Technická univerzita v Liberci
Fakulta strojní

DIPLOMOVÁ PRÁCE

Informační systém pro správu skladového hospodářství

Liberec 2003 Martin Hudík

Technická univerzita v Liberci

Fakulta strojní

Studijní obor : 23 – 40 – 8 Automatizované systémy řízení ve strojírenství
Zaměření : Automatizace inženýrských prací

Katedra aplikované kybernetiky

Informační systém pro správu skladového hospodářství

Information system for stock control administration

Martin Hudík

Vedoucí diplomové práce : Ing. Jan Žitník
Konzultant diplomové práce : Ing. Jan Žitník

Rozsah diplomové práce:

Počet stran: 46
Počet příloh: 2

ANOTACE

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní
Katedra aplikované kybernetiky

Studijní obor :	23 – 40 – 8 Automatizované systémy řízení ve strojírenství
Studijní zaměření :	automatizace inženýrských prací
Diplomant :	Martin Hudík
Téma práce :	Informační systém pro správu skladového hospodářství
Theme of work :	Information system for a stock control administration
Rok obhajoby DP :	2003
Vedoucí DP :	Ing. Jan Žitník
Konzultant :	Ing. Jan Žitník

Anotace : Cílem této diplomové práce je vytvoření jednoduchého informačního systému pro správu skladového hospodářství pro menší distibuční firmu, který bude obsahovat evidenci zboží, zákazníků a skladových dokumentů a export do aplikace Účto 2002.

Annotation : The purpose of this diploma work is to create simple information system for stock control administration for smaller distribution firm, that should include goods, costumer and stock documents evidence and export to Účto 2002 application.

Poděkování

Chtěl bych poděkovat všem, kteří mi pomáhali, či mě jinak podporovali, při zpracovávaní této diplomové práce a to hlavně mým rodičům, mému bratrovi, mým přátelům a kamarádům.

Dále bych chtěl poděkovat vedoucímu diplomové práce Ing. Janu Žitníkovi zvláště za cenné postřehy při dokončování aplikace.

Místopřísežné prohlášení

„Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury.“

V Liberci 23. 5. 2003

…….
Martin Hudík

Seznam použitých zkratek

ADO	ActiveX Data Objects
BCL	Base Class Library
CLR	Common Language Runtime
CSS	Cascading Style Sheets
DSSSL	Document Style Semantics and Specification Language
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
MSDE	Microsoft Data Engine
MS-DOS	Microsoft Disk Operating System
MSIL	Microsoft Intermediate Language
ODBC	Open Database Connectivity
PDA	Personal Digital Assistent
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SQLOLEDB	SQL Server OLE DB Provider
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Obsah

1 Úvod

Jednouživatelský informační systém pro správu skladového hospodářství je určen pro menší distribuční firmu zabývající se redistribucí dentistického materiálu od výrobce ke konečnému zákazníku. Měl by nahradit stávající, již nevyhovující, informační systém a zabezpečit export některých ze zaznamenaných dokumentů, jako jsou faktury, či evidence odběratelů a dodavatelů.

Řešení této diplomové práce se pak skládalo z těchto hlavních kroků: seznámení se s problematikou skladového hospodářství, analýza problému, návrh vhodného řešení, volba vhodných nástrojů pro realizaci, seznámení se s vývojovým prostředím platformy Microsoft, v míře, umožňující navržení a realizaci předmětu práce-informačního systému.

1.1 Charakteristika problémové oblasti

Jádrem informačního systému společnosti je obvykle vedle její finanční části, část určená pro řízení skladového hospodářství. Jedná se o soubor činností spojených s uspokojování hmotných potřeb odběratelů. Zdrojem pro zajištění potřeb je kompletace dodávek obchodního zboží, jednak ze skladovaných zásob a nebo z dodatečně objednaných nákupů od dodavatelů. Informace souvisící s logistickým zajištěním/naplněním, řeší podsystém poskytující podklady a evidenci pro irelevantní rozhodovací procesy.

1.2 Předpoklady a omezení práce

Diplomová práce si klade za cíl, dle zadání, řešit podsystém jednouživatelsky, na technickém prostředku představující osobní počítač umožňující provoz operačního systému Windows NT/2000/XP s dostupem na tiskárnu. Vzhledem k pojetí úkolu a časových možností řešitele není uvažována portace do jiných prostředí. Naopak otevřeným se jeví rozvoj podsystému směrem k dalšímu propojení na související podsystémy z oblasti

Účetnictví. Rovněž atributem řešení je otevřenost k eventuálnímu rozvoji pro vzdálený přístup k informacím s oblasti skladového hospodářství via INTRANET/INTERNET.

1.3 Použité postupy a metody

V první fázi řešení úkolu tj. v analýze jsem vycházel z ukázky funkce současného morálně zastaralého programového vybavení v MS-DOSu. Dalším vodítkem mi byly konzultace o provozu programu a zejména o potřebách informací pro řízení logistických toků. Výsledkem analýzy byly návrhy procesů, datové struktury a návrhy obrazovek. Posléze jsem zvolil programovací nástroje a výslednou architekturu řešení, za přihlédnutí především k sortimentu skladu, k frekvenci pohybů na skladě, k požadované dostupnosti systému a k technickým a systémovým možnostem zadavatele a řešitele.

2 Popis objektu a analýza procesů

2.1 Teoretický model

Z kybernetického hlediska je obecně otevřený systém zjednodušeně určen vstupy, výstupy a chováním.

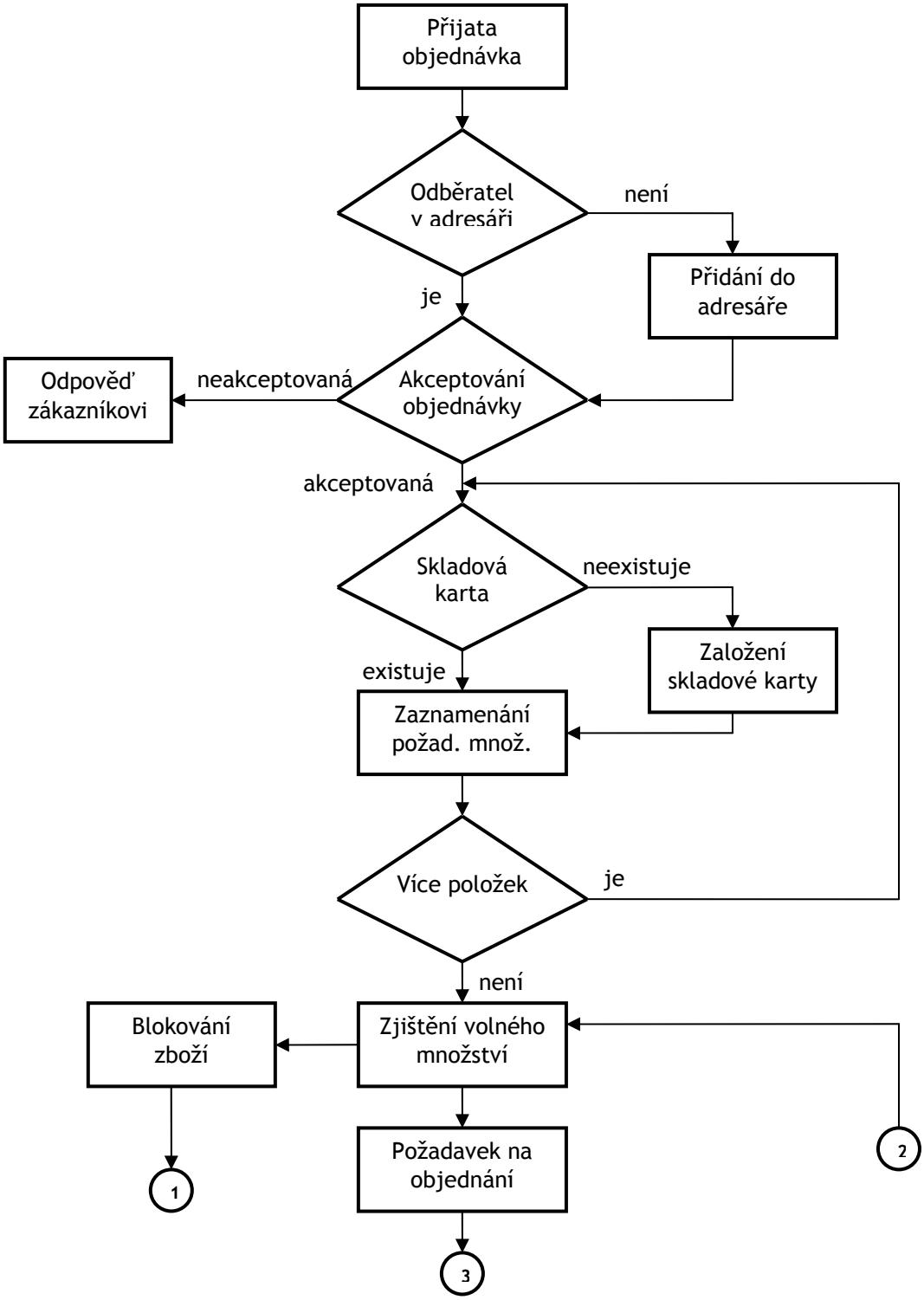
V našem případě, v rovině hmotných toků, jsou vstupy dodávky zboží od dodavatelů do skladu a výstupy následující dodávky odběrateli. Cílovým chováním skladu je navigování zboží jedním směrem dodavatel -> sklad -> odběratel (obr.2.1). Naproti v rovině informací směřují informace zprvu opačně odběratel -> sklad -> dodavatel (obr.2.2) ve fázi inicializační a ve fázi realizační ve shodě s hmotným tokem tj. dodavatel -> sklad -> odběratel (obr.2.1). Chování skladu je pak skladování zboží pro funkci vyrovnávací (bufferu) k snížení frekvence dodávek a zejména pro zajištění schopnosti maximální včasnosti dodávek, významnou pro konkurenceschopnost zajišťovatele. S tohoto pohledu je požadavek na chování skladu mít co největší zásoby s co nejširším sortimentem. Protikladným, ekonomickým - podstatě regulačním hlediskem je požadavek na skladované množství a sortiment blížící se nulovým hodnotám, ve snaze minimalizovat vázaný kapitál v zásobách. Výsledným chováním je hledání optima mezi uvedenými protiklady, pro kterýž je záměr vytvořit programové vybavení práce.

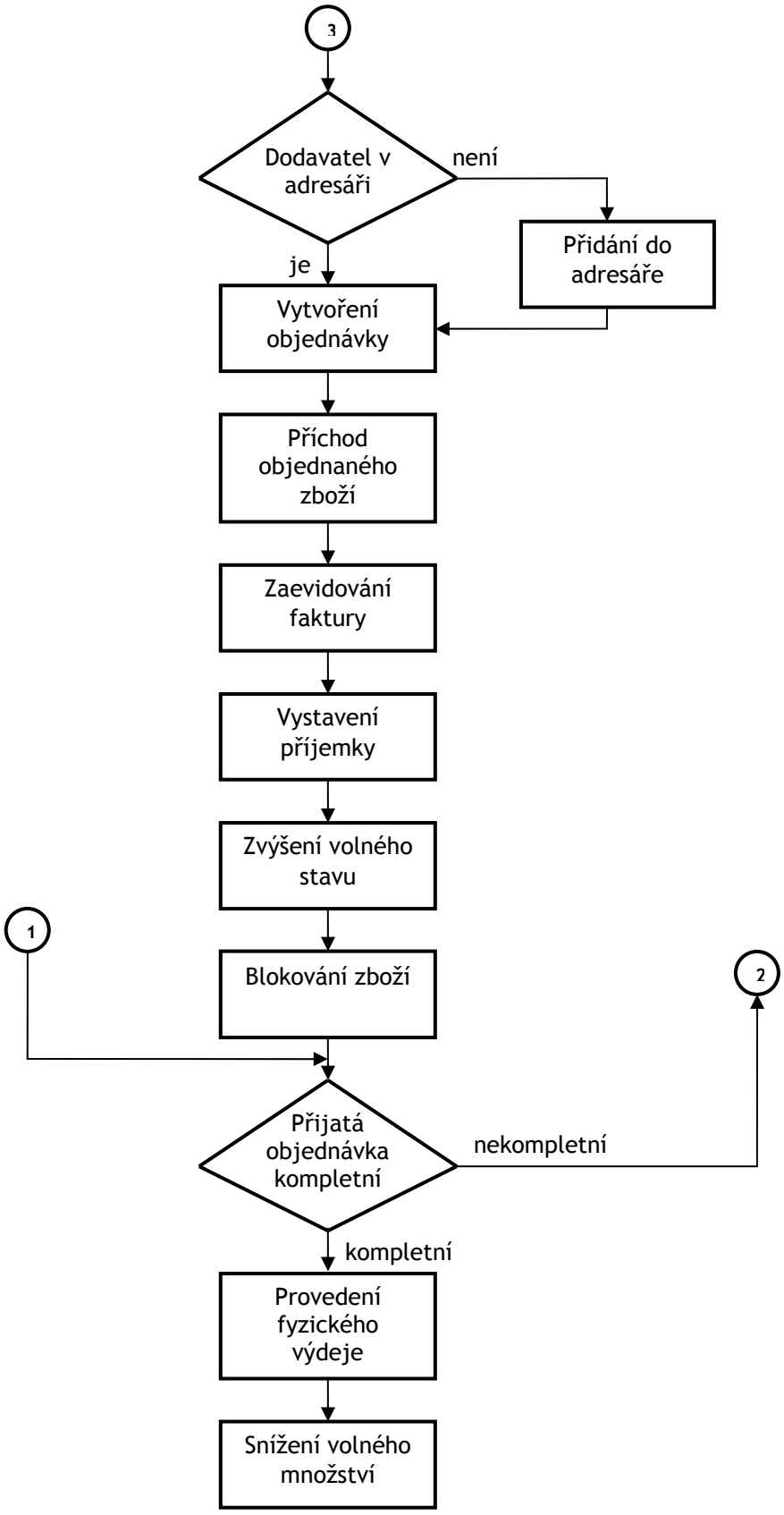
Obr. 2.1 Tok informací ve fázi inicializační

Obr. 2.2 Hmotný tok zboží, tok informací ve fázi realizační

2.2 Související funkce a procesy

2.2.1 Princip vyřízení přijaté objednávky - graficky





Obr. 2.3 Princip vyřízení přijaté objednávky

2.2.2 Princip vyřízení přijaté objednávky - slovně

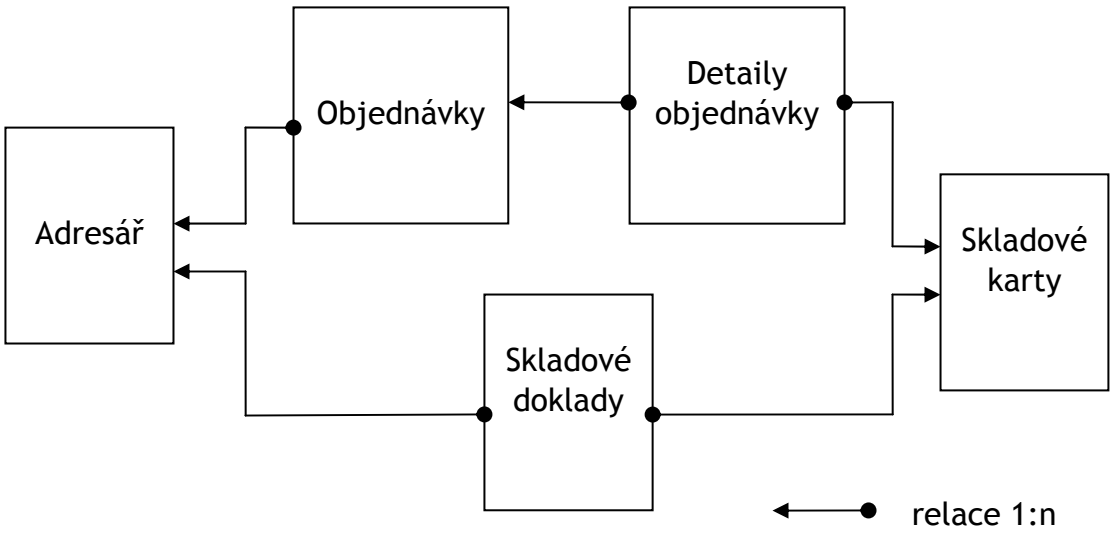
1. Přijatá objednávka na zboží je zaevidována a opatřena pořadovým evidenčním číslem.
2. Zjištění, zda je odběratel v adresáři
    a. pokud není, zaevidování odběratele a všech náležitostí, pak zajištění odkazu na adresu
    b. pokud je - zajištění odkazu na adresu
3. Objednávka je posouzena :
    a. neakceptovaná, odpověď zpět odběrateli
    b. akceptovaná -> vyřízení
4. Zjištění zda existuje karta požadovaného zboží
    a. neexistuje - založení karty a zajištění odkazu na kartu
    b. existuje - zajištění odkazu kartu
5. Zaznamenání požadovaného množství.
6. Pokračování v cyklu v případě více položek/řádků objednávky od bodu 4
7. Zjištění požadovaného množství a volného množství zboží na skladě
    a. pokud je požadované množství menší, či rovno volnému množství – blokace ve prospěch objednávky
    b. pokud je požadované množství větší – blokace odpovídacího množství, rezervace zbytku a vydání požadavku na objednání
8. Příjem požadavku na objednání
9. Zjištění zda existuje v adresáři odpovídající dodavatel
    a. neexistuje - založení jeho specifikace, odkaz na dodavatele
    b. existuje, odkaz na dodavatele
10. Zjištění všech nových požadavků na objednání zboží, vytvoření objednávky a připravením k odeslání
11. Příchod zboží do firmy
12. Změna stavu objednávky na „vybavená“.
13. Zaevidování přijaté faktury, zajištění odkazu na dodavatele, po jejím splacení změna stavu na „splacená“.
14. Vystavení příjemky na základě jednotlivých položek dodacího listu v cyklu
15. Zvýšení volného stavu na skladu o odpovídající množství

16. Blokace rezervovaného zboží z volného množství.
17. Zjištění, zda je přijatá objednávka kompletní
    a. ano - změna stavu na „vybavená“ a vytvoření odpovídající faktury, dodacího listu a výdejky.
    b. ne – návrat k bodu 7.
18. Provedení fyzického výdeje, snížením disponibilního množství.
19. Po fyzickém předání zboží odběrateli změna stavu dodacího listu na „realizovaný“
20. Po splacení vystavené faktury odběratelem změna stavu faktury na „splacená“

2.3 Návrh datového modelu.

2.3.1 Vývoj návrhu datového modelu

Návrhů struktury databáze a datových toků bylo při zpracovávání mnoho, resp. na počátku byla jedna jednoduchá struktura s několika málo tabulkami, která se postupně, podle potřeb, měnila a hlavně rozrůstala. Šlo o tabulku s evidencí odběratelů a dodavatelů, tabulku objednávek, detailů objednávek, tabulku zboží na skladě a skladových dokladů vzájemně propojených dle obrázku:



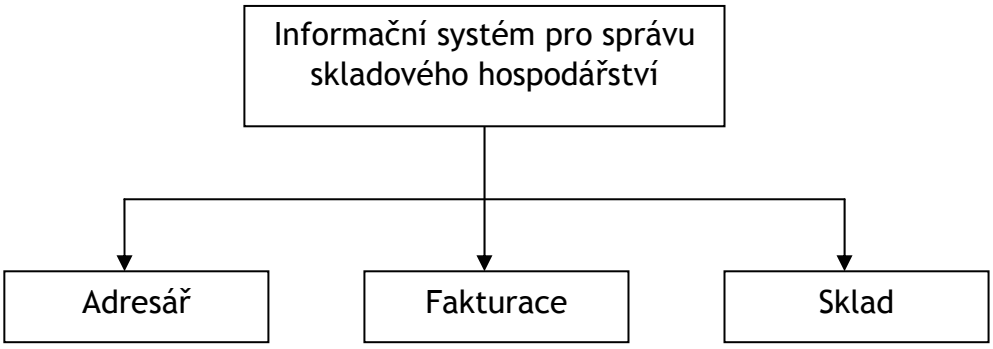
Obr. 2.4 Počáteční návrh struktury databáze

Tato koncepce se postupně rozrostla z těchto pěti tabulek na strukturu čítající tabulek šestnáct (viz. dále).

2.3.2 Funkční analýza

Celý systém lze logicky rozdělit na tři základní části:

- část evidence odběratelů a dodavatelů – adresář
- část evidence dokumentů – fakturace
- skladová část – sklad



Obr.2.5 Funkční schéma databáze

Tyto tři části jsou však úzce provázány a navenek, tvoří komplexní celek, kde každá část má svou důležitou a nezastupitelnou úlohu, bez níž by systém nefungoval.

Adresář

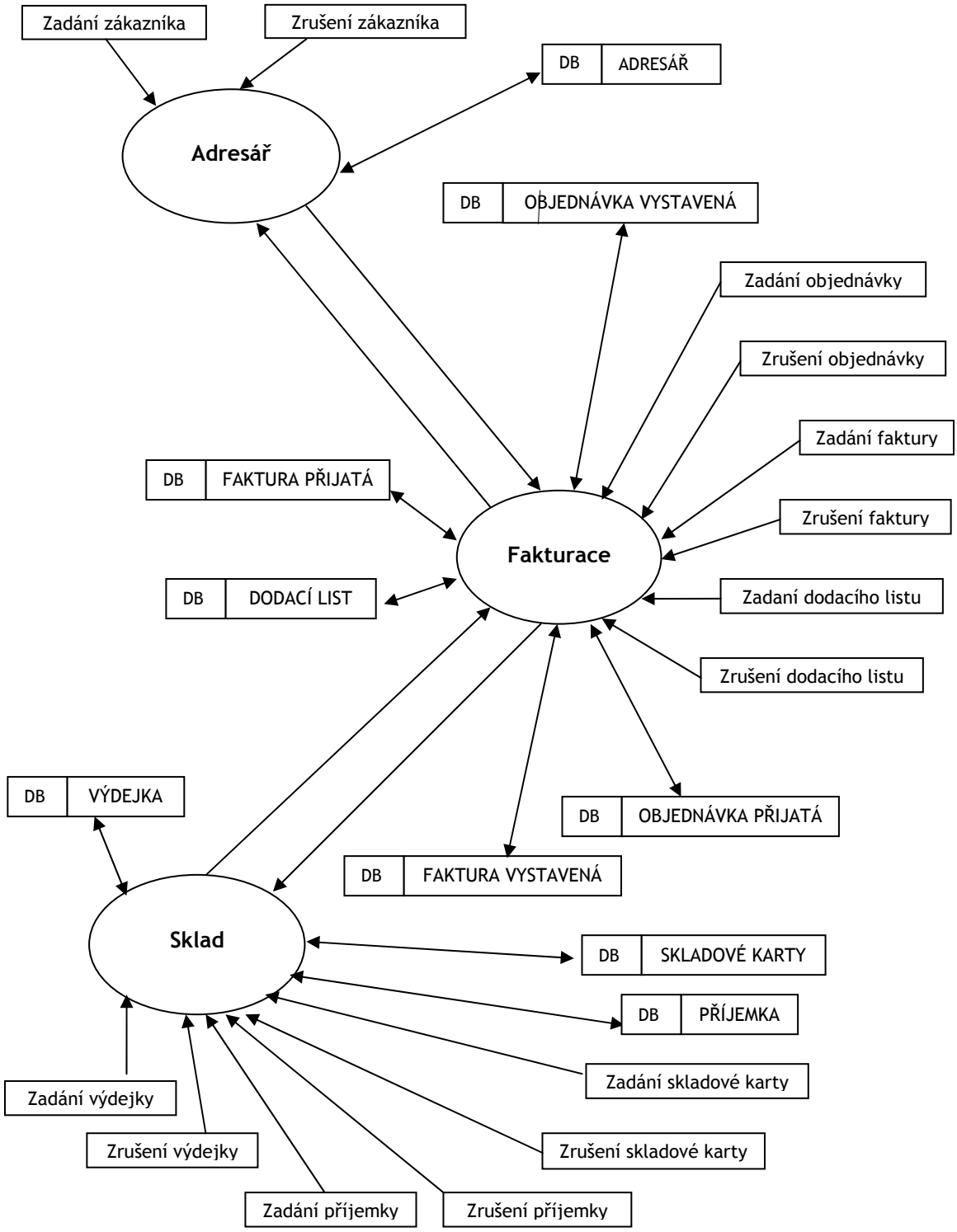
Jak bylo již zmíněno, adresář je kompletní evidencí dodavatelů a odběratelů a všech potřebných údajů o nich jako jsou např. adresa, IČ, DIČ, telefonní číslo, fax, e-mailovou adresu, banka, účet, slevy,…

Fakturace

Tato část je evidencí dokumentů jako jsou přijaté a vystavené objednávky, přijaté a vystavené faktury a dodacích listů. Obsahuje všechny důležité údaje objednávek, faktur resp. dodacích listů včetně stavu jednotlivých dokumentu, tedy zda-li je objednávka nevybavená, či již vybavená, zda je faktura splacená, či nesplacená a zda je dodací list realizovaný nebo ještě nerealizovaný. Seznamy těchto dokumentů lze i podle těchto stavů filtrovat a tisknout.

Sklad

Tato část obsahuje jednak skladové doklady, tedy příjemky a výdejky, skladové karty a pak seznamy zboží roztříděné podle různých filtrů tedy zboží, které je blokováno, rezervováno, objednáno a nutno objednat, aktuální stav skladu, skladové obraty za určité období a pohyby na skladě.



Obr. 2.6 Základní diagram datových toků

2.3.3 Definice datových zásobníků

Datovými zásobníky lze nazvat jednotlivé tabulky databáze uchovávající odpovídající údaje. Následuje výpis všech tabulek s jednotlivými poli, typem pole a označením, zda-li jde o primární klíč tabulky:

adresar	(adresar)
adresarid	int
nazev	string
adresa	string
mesto	string
psc	int
ico	string
dic	string
kontakt	string
telefon	string
fax	string
mobil	string
email	string
banka	string
ucet	string
splatnost	short
dph	string
sleva	float
poznamky	string
typ	string

Tabulka údajů o zákaznících

skladkarty	(skladkarty)
skladid	int
nazev	string
mj	string
dph	short
skryt	boolean
minimum	float
poznamky	string
cenap	decimal
cenan	decimal

Tabulka skladových karet

objednavkavys	(objednavkavys
objednavkaid	int
adresarid	int
popis	string
poznamky	string
stav	string
denid	int
mesicid	int
rokid	int
skryt	boolean
vystavilid	int

Tabulka vystavených objednávek

polozkyvys	(polozkyvys)
polozkyid	int
pocet	float
skladid	int
objednavkaid	int
sleva	float
cena	decimal

Tabulka položek přijaté objednávky

objednavkaprij	(objednavkaprij
objednavkaid	int
adresarid	int
objodbid	string
popis	string
poznamky	string
stav	string
denid	int
mesicid	int
rokid	int
skryt	boolean
vystavil	string

Tabulka přijatých objednávek

polozkyprij	(polozkyprij)
polozkyid	int
pocet	float
skladid	int
objednavkaid	int
cena	decimal
sleva	float
rezervovano	float

Tabulka položek přijaté objednávky

fakturaprij	(fakturaprij)
fakturaid	int
adresarid	int
denvid	int
mesicvid	int
rokvid	int
densid	int
mesicsid	int
roksid	int
denuzpid	int
mesicuzpid	int
rokuzpid	int
skryt	boolean
vystavil	string
doprava	string
uhrada	string
konsts	string
variabs	string
specifs	string
popis	string
poznamky	string
objednavkaid	int
fakturadid	string
stav	boolean

Tabulka přijatých faktur

fakturavys	(fakturavys)
fakturaid	int
adresarid	int
densid	int
mesicsid	int
roksid	int
denvid	int
mesicvid	int
rokvid	int
denuzpid	int
mesicuzpid	int
rokuzpid	int
skryt	boolean
doprava	string
uhrada	string
popis	string
poznamky	string
vystavilid	int
objednavkaid	int
objodbid	string
konsts	string
specifs	string
variabs	string
stav	boolean

Tabulka vystavených faktur

blokace	(blokace)
polozkyid	int
dokladid	int
pocet	float

Tabulka blokací

vystavil	(vystavil)
vystavilid	int
jmeno	string
telefon	string
fax	string
email	string

Tabulka vystavitelů

mesic	(mesic)
mesicid	int
mesic	short

Pomocná tabulka měsíců

rok	(rok)
rokid	int
rok	short

Pomocná tabulka roků

dodacilist	(dodacilist)
dodacilistid	int
adresarid	int
denvid	int
mesicvid	int
rokvid	int
skryt	boolean
vystavilid	int
doprava	string
popis	string
poznamky	string
specifs	string
variabs	string
objednavkaid	int
stav	boolean
objodbid	string

Tabulka dodacích listů

den	(den)
denid	int
den	short

Pomocná tabulka dnů

prijemka	(prijemka)
dokladid	int
fakturaid	int
adresarid	int
skladid	int
sarze	string
pocet	float
cena	decimal
skryt	boolean
vystavil	string
denid	int
mesicid	int
rokid	int
popis	string
poznamky	string

Tabulka příjemek

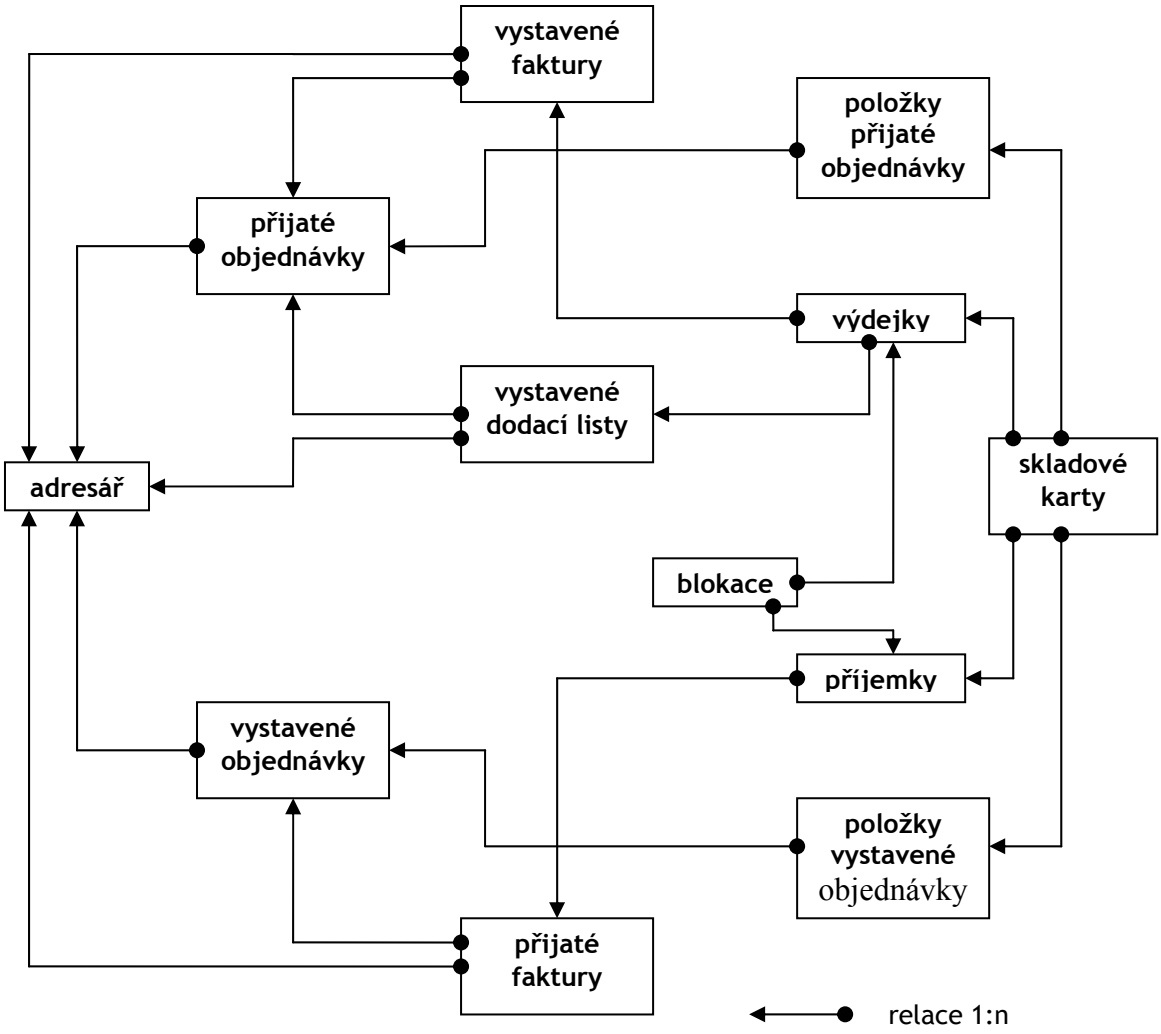
vydejka	(vydejka)
dokladid	int
dodacilistid	int
fakturaid	int
adresarid	int
skladid	int
sarze	string
pocet	float
cena	decimal
skryt	boolean
vystavil	string
denid	int
mesicid	int
rokid	int
popis	string
poznamky	string

Tabulka výdejek

2.3.4 Relační datový model

Relace znamená nějaký určitý vztah a relační datový model pak udává vztahy mezi jednotlivými tabulkami, mezi jednotlivými daty.

Toto je zjednodušený relační diagram:



Obr. 2.7 zjednodušený relační diagram

3 Tvorba uživatelského rozhraní

Uživatelské rozhraní bylo voleno tak, aby bylo jednoduché a intuitivní.

Hlavní obrazovka aplikace je složena ze čtyř základních ovládacích prvků: Tree View – strom, Main Menu – hlavní menu aplikace, List View – seznam a Tool Bar – panel s tlačítky.

Výběr může být prováděn, buď přes ovládací prvek Tree View nebo přes hlavní menu aplikace. Příslušná data jsou pak zobrazována v jediném ovládacím prvku List View, který je dynamicky přizpůsobován typu zobrazovaných dat. Pomocí panelu s tlačítky, stejně jako pomocí hlavního menu lze přidat novou položku, prohlížet detaily položky, upravovat položku, odebírat položku, provádět tisk, či ukončit běh aplikace.

Přidávání položek, prohlížení detailů, úprava, tisk jsou pak prováděny přes odpovídající dialogová okna.

4 Řešení exportu dat

Export evidence odběratelů a dodavatelů a evidence faktur do účetního programu Účto 2002 je řešen zapsáním těchto dat do textového souboru, s kódováním Latin II a v daném formátu. Soubor se vytvoří implicitně v adresáři aplikace.

Do textového souboru se vždy zapíší všichni odběratelé a dodavatelé, resp. faktury a program Účto 2002 pak importuje jen nové položky.

Export lze provést pomocí položky v menu „Soubor -> Export“.

5 Volba vývojového prostředí

5.1 Správa a analýza dat

Výběr správného nástroje pro správu a analýzu dat úzce souvisí s typem a použitím výsledné aplikace, respektive počtem připojených uživatelů, tedy jestli jde o síťovou, víceuživatelskou, aplikaci, či službu nebo o aplikaci, kterou bude používat jeden uživatel popřípadě omezený počet uživatelů.

Pro síťové aplikace by byl výhodný nějaký robustní, profesionální, rychlý, ale také složitý, náročnější na systémové prostředky a na správu a v mnoha případech i finančně náročnější, typu Microsoft SQL, MySQL, DB/2, Oracle,…

V tomto případě jde o aplikaci, kterou bude využívat pouze jeden uživatel, takže zcela postačuje jednodušší, univerzální, levnější a pro případ jednoho uživatele i rychlejší nástroj – Microsoft Access 2002 z kancelářské sady Microsoft Office XP.

5.2 Nástroj pro vytvoření vlastní aplikace

Možnost výběru programovacího jazyka byla zúžena zadáním diplomové práce na jazyky platformy .NET Framework, tedy prakticky na jazyky Visual C++ .NET, Visual Basic .NET a Visual C#. NET.

Visual C++ .NET

C++ je moderní objektově orientovaný jazyk založený na jazyku C. Je to výkonný, flexibilní a efektivní programovací jazyk. Vysoký výkon a možnosti jemného řízení umožňují vytvářet aplikace, které jsou spouštěny jako součást operačního systému. vhodný zejména pro tvorbu rozsáhlých programů, avšak pro menší a jednodušší projekty je jeho použití značně neefektivní.

Visual Basic .NET

Oproti předchozí verzi (Visual Basic 6.0) zcela přepracovaný programovací jazyk, který nyní dává k dispozici i pokročilé technologie, jako je objektově orientované programování s podporou dědičnosti a strukturované ošetřování výjimek. Je relativně jednoduchý pro naučení a zdrojový kód je snadno čitelný. Umožňuje snadné vytváření webových služeb XML a aplikací pro platformu .NET.

Visual C# .NET

Zcela nový programovací jazyk vycházející z jazyka C, resp. C++, objektově orientovaný, který se snaží přiblížit výkonnosti a flexibilitě jazyka C++ a jednoduchosti čtení a zápisu a produktivitě jazyka Visual Basic .NET. Umožňuje provádět "neřízená" volání (volání kódu spouštěného mimo rámec .NET Framework). Je předurčen k vytváření mocných aplikací a webových služeb XML pro platformu .NET.

Vzhledem k výše uvedenému a díky mé touze se poznat nové technologie jsem pro realizaci daného úkolu zvolil programovací jazyk Visual C# .NET.

5.3 Propojení aplikace s databází

Volba technologie pro přístup k datům a poskytovatele dat byla prakticky provedena už volbou nástroje pro správu a analýzu dat a programovacího jazyka a nakonec i zvolením platformy Microsoft .NET, kde přístup k datům a manipulaci s nimi je úzce provázán na knihovnu ADO. NET. Jako poskytovatel dat je pak, vzhledem k volbě Microsoft Access 2002 jako nástroje pro správu dat, zvolen OLE DB .NET Data Provider.

6 Teoretické vlastnosti vývojového prostředí

6.1 Platforma Microsoft .NET

6.1.1 Obecný popis

Platforma Microsoft .NET. je sada vývojových nástrojů a operačních systémů k tvorbě, nabídce a využití webových XML služeb. XML je akronymem pro eXtensible Markup Language - univerzální formát pro strukturované dokumenty a data na síti Internet. XML představuje standardní protokol spravovaný organizací World Wide Web Consortium a je klíčovou technologií architektury Microsoft .NET.

Microsoft .NET se od strategií jiných významných firem příliš neliší. Microsoft nebyl ani první a nebyl a ani řešení není jedinečné. Podobná řešení, i když v různé fázi rozpracovanosti, nabízí prakticky všechny velké firmy. Přesto se Microsoft .NET v něčem liší. Stal se prvním řešením, které vyvolalo skutečně masový zájem o problematiku webových služeb. Kdykoliv, kdekoliv a z libovolného zařízení dává možnost přístupu k informacím, datům anebo možnost ovládat zařízení připojená k celosvětové informační síti ve svém domě či bytě. Ať už to je vzdálený přístup, např. z pracovní cesty, k datům a informacím na počítači v zaměstnání, objednání lístků na koncert oblíbené skupiny z obývacího pokoje, nastavení nahrávání oblíbeného pořadu na video, či spuštění topení ještě před příchodem domů.

Platforma, přístup, architektura Microsoft .NET také mění způsob, jakým nahlížíme na počítače a jak je využíváme. Až dosud ve výpočetní technice dominovala dvě pojetí: serverové a desktopové. Microsoft .NET je distribuovaná výpočetní forma, která nepoužívá tradiční rozlišování na desktopy a servery. Distribuovaná výpočetní technika je programovací model, u kterého dochází ke zpracování dat na mnoha různých místech (uzlech) sítě. Zpracování se provádí vždy tam, kde je to nejúčelnější, kde to má větší smysl: na serveru, webovém serveru, v osobním počítači, kapesním zařízení nebo v

jiném inteligentním zařízení. Tento aplikační model je účinnější než model koncentrace kolem serveru a snižuje možnost vzniku "úzkého profilu" toku dat.

6.1.2 .NET a standardy

Jak je u webových služeb běžné, je platforma .NET založena na otevřených standardech (XML, HTTP, SOAP,...) tak, aby mohla fungovat na všech platformách. Zaměřena přitom je na vývoj, nasazení, provozování a používání nové generace distribuovaných síťových služeb a aplikací.

Významnou úlohu v rámci platformy .NET hraje především jazyk XML, který zajišťuje výměnu dat mezi jednotlivými aplikacemi na internetu. Ten umožňuje provázat aplikace, u nichž je vzájemné předávání dat za běžných okolností velmi nesnadné. Aplikace jsou konstruovány pomocí řady webových služeb, které poskytují data a služby jiným aplikacím. Výměna dat přitom může probíhat nejen mezi jednotlivými aplikacemi, ale i mezi různými zařízeními připojenými k síti. A protože obecný model pro tvorbu aplikací představují webové služby, mohou být použity na libovolném operačním systému.

6.1.3 Přínosy platformy .NET

Práce s platformou .NET je mnohem více osobní, integrují se zde výpočetní interakce koncových uživatelů s využitím připojených webových XML služeb, poskytovaných novou generací inteligentních zařízení. Z technického hlediska je práce s platformou .NET kombinací webových XML služeb a tam, kde je to vhodné i lokálního aplikačního kódu.

Výsledkem jsou robustní a bezpečné webové služby založené na XML, webové formuláře, aplikace s grafickým uživatelským rozhraním Win32, konzolové aplikace, nejrůznější nástroje i samostatné komponenty.

6.1.4 .NET Framework

.NET Framework je vlastně základ celé technologie .NET, základní jádro pro tvorbu, nasazení, spouštění a provoz webových XML služeb a dalších aplikací. Je to sada objektů a šablon, která obsahuje mimo jiné CLR (Common Language Runtime), který zajišťuje překlad řízeného kódu a je potřebný pro běh všech aplikací .NET, třídy Framework - Base Class Library (BCL), či ASP.NET. Díky .NET Framework získávají aplikace mnoho nových vlastností jako je například bezpečnost. Doprovodná infrastruktura, .NET Compact Framework, je sada programovacích rozhraní umožňujících vývojářům zaměřit se na mobilní zařízení, jako jsou inteligentní (smart) telefony a PDA (Personal Digital Assistent).

Kompilátory pro .NET vytváří (nebo dokáží vytvořit) tzv. řízený kód (managed code). Tento kód si lze představit jako kód interpretových jazyků. Řízený kód se kompiluje do nativního kódu až v době spuštění nebo instalace softwaru. Tím je běh aplikace samozřejmě zpomalen, ale na druhou stranu může být optimalizován pro konkrétní typ procesoru (to právě zařídí .NET Framework, který musí být instalován na cílovém PC). Lze dokonce použít tzv. Just-in-Time kompilace, při které se kompiluje jen to, co je právě potřeba a pokud dochází paměť jsou nepotřebné funkce odstraněny. Při dalším volání je nutné funkce znovu překompilovat.

Z důvodu přenositelnosti kódu na jiné platformy se kód kompiluje nejprve do MSIL (Microsoft Intermediate Language). MSIL je procesorově nezávislý jazyk podobný assembleru. Program (spustitelný program) není tvořen jen kódem MSIL, ale obsahuje jakousi hlavičku, která se nazývá Manifest. Manifest a MSIL kód nazýváme Assembly. Assembly může být spustitelný program .exe nebo knihovna .dll. Může navíc obsahovat další soubory jako jsou zdroje. V Manifestu jsou obsažena metadata, která zajišťují Assembly mnoho užitečných vlastností (správa verzí apod.). Pokud chceme program spustit, spustí se CLS, zkompiluje jazyk MSIL do přirozeného kódu a program se spustí. Je to velmi podobné například systému Java Virtual Machine. Další důležitá změna je to, že již nepotřebujeme registr. Díky metadatům, které se

produkují při kompilaci do MSIL, ví CLR, jak aplikaci správně spustit. Vše se ukládá přímo do aplikace. Kompilací kódu se samozřejmě zvýší rychlost zpracování.

Pro napsání zdrojového kódu lze použít libovolný jazyk, který podporuje MSIL. Za tímto účelem byly vytvořeny další jazyky, například jazyk Visual C# .NET, Visual Basic .NET nebo Visual C++ .NET. Ale pokud máme kompilátor např. Pascalu, který vytváří řízený kód, můžete tento jazyk použít pro tvorbu .NET aplikací (teoreticky to jde u libovolného programovacího jazyka).

6.2 XML

6.2.1 Úvod

Jak bylo již zmíněno, XML (eXtensible Markup Language) je jakousi „páteří“ architektury platformy Microsoft .NET. Patří do rodiny značkovacích jazyků a byl vytvořen jako způsob pro strukturalizaci, ukládání a posílání informací. V současné době se o XML hovoří nejčastěji v souvislosti s Webem a považuje se za nástupce dnes používaného jazyka HTML, ale možnosti využití jsou mnohem širší. Dokumenty XML lze rozšiřovat, aby mohly obsáhnout více informací.

6.2.2 Využití XML

O XML se nejčastěji hovoří jako o novém jazyku pro tvorbu webových stránek. Situace se dnes vyvíjí tak, že nám XML umožní rozšiřovat množinu elementů, které nám při tvorbě stránek nabízí HTML. Stránky v XML jsou zároveň snazší pro čtení než ty současné v HTML, které obsahují mnoho chyb. To umožní vývoj nových jednoduchých prohlížečů určených zejména pro různá kapesní mobilní zařízení.

XML však nezůstává pouze technologií určenou pro Web. Využití nalezne všude, kde je potřeba jedny informace prezentovat v několika formátech.

Výhoda XML spočívá v tom, že kromě samotného textu nese i informaci o jeho významu. Konverze do libovolného formátu je pak snadná a může probíhat zcela automaticky. XML proto nalézá uplatnění při tvorbě technické dokumentace, což v některých oblastech průmyslu znamená práci s tisícistránkovými dokumenty, které je potřeba několikrát ročně aktualizovat a distribuovat uživatelům v různých formách.

Mezi prvky XML existují vazby. Prvky jsou ve vztahu struktur rodič a potomek. Prvek XML je všechno od počátečního tagu prvku do ukončujícího tagu prvku.

Prvek může mít:

- prvkový obsah,
- smíšený obsah,
- jednoduchý obsah,
- prázdný obsah,
- atributy.

Největší objem dat, který bude v podobě XML přenášen, budou strukturovaná data, která se dnes ukládají do relačních databází. XML poslouží jako vhodný přenosový formát při komunikaci mezi aplikacemi různých výrobců, mezi webovým serverem a prohlížečem apod.

XML má všechny předpoklady stát se opravdu revolučním prostředkem pro popis, ukládání, přenos a publikaci informací.

6.2.3 XML jako nástupce HTML

Výhoda XML spočívá v tom, že autor stránky může používat vlastní tagy, které dokáží mnohem přesněji označit význam prezentovaných informací. To má velký význam především pro vyhledávání informací. Dnešní Web je informacemi přehlcen a nalézt konkrétní informací je stále obtížnější a mnohdy i nemožné. Tento problém nevyřeší sebelepší prohledávací servery,

pokud jim nepomohou autoři stránek, kteří pomocí XML uloží do stránek mnohem více metainformací.

Dokumenty nejen prohledáváme, ale v konečné fázi i zobrazujeme. V jazyce HTML je vše jednoduché - jednotlivé tagy určují, jak se má daná část stránky zobrazit. V XML žádná podobná přímá vazba neexistuje, a proto musíme způsob zobrazení jednotlivých elementů dokumentu definovat jiným způsobem - pomocí stylů. Ve stylu pak můžeme pro každý element určit, jak se má zobrazit (font a velikost písma, barva, zarovnání, umístění apod.).

Velká výhoda stylů spočívá v tom, že jsou od samotného dokumentu odděleny a jsou v samostatném souboru. Jeden styl můžeme využívat v mnoha XML-dokumentech a dosáhnout tak jednotného vzhledu mnoha stránek. Na druhou stranu, pokud potřebujeme změnit design stránky, stačí drobně opravit styl a změna se promítne na všech stránkách, které styl používají.

Kaskádové styly (CSS) bylo možné používat i v HTML. Jejich použití však bylo pouze doplňkem, kterým jsme mohli modifikovat vzhled některých elementů. Autoři stránek mohli zcela beztrestně "zneužívat" jednotlivé HTML tagy pro dosažení požadovaného vzhledu stránky. XML-dokument bez stylu nelze zobrazit, autoři tak ztratí důvod pro vytváření špatně strukturovaných stránek, které však dosahují požadovaného vzhledu.

6.2.4 Zobrazení XML

Vzhled jednotlivých elementů a tím i celého dokumentu je potřeba definovat pomocí stylu. Pro definici vzhledu XML dokumentů se dnes používají tři stylové jazyky CSS, XSL a DSSSL.

CSS (Cascading Style Sheets) není nic jiného než kaskádové styly používané v HTML. Kaskádové styly byly původně určeny pouze pro prezentování informací na obrazovce. Další parametry formátování, které je potřeba zadat pro tištěný výstup, byly přidány do nové verze kaskádových stylů CSS2. Pro potřeby XML byly do CSS2 přidána i další rozšíření, která umožňují konkrétní

styl aplikovat i na základě hodnoty atributu. CSS1 umožňovalo styl aplikovat pouze na určitý element bez návaznosti na obsah jeho atributů.

Protipólem k CSS je jazyk DSSSL (Document Style Semantics and Specification Language). DSSSL je rozsáhlý jazyk, který byl původně vyvinut pro použití s SGML. Jeho použití s XML však nic nebrání. DSSSL je tak komplexní jazyk, že jej zatím nepodporuje příliš mnoho aplikací. Pro potřebu elektronického publikování byla definována jeho podmnožina DSSSL-O, která se hodí pro méně náročné aplikace.

Zlatou střední cestou mezi oběma jazyky je XSL (eXtensible Stylesheet Language). Tento jazyk má možnosti srovnatelné s DSSSL, jeho syntaxe je však mnohem jednodušší. Zajímavostí je, že XSL se zapisuje v jazyce XML. Pro tvorbu stylů můžeme tedy použít stejný editor, který používáme pro tvorbu dokumentů.

6.3 Programovací jazyk Visual C# .NET

6.3.1 Úvod

Programovací jazyk Visual C# .NET by se měl stát hlavním vývojovým prostředkem pro platformu .NET. Jak je z názvu patrné, tak se hlásí k dědictví jazyka C/C++ resp. Visual C++, tedy by měl být tak výkonný a flexibilní jako ekvivalent napsaný v C++, ale jednoduchostí psaní a čtení programového kódu by se měl blížit zase prostředí Visual Basic. Jako hlavní pole působnosti pro nový jazyk je zvolen vývoj XML síťových aplikací právě pro platformu .NET. K hlavním vlastnostem jazyka patří automatická správa paměti přes tzv. garbage collection, automatická inicializace proměnných, typová bezpečnost, výrazné potlačení pointrů, silná orientace na objekty a COM vůbec.

6.3.2 Hlavní rysy jazyka Visual C# .NET

Jazyk Visual C# .NET (dále jen C#) je relativně jednoduchý, moderní, flexibilní, objektově orientovaný programovací jazyk. Jeho velkou výhodou je samozřejmě to, že je nový, a proto moderní rysy jsou hladčeji implementovány, než u starších programovacích jazyků, kde musejí být často složitě dodatečně vnášeny.

Při úplně prvním pohledu se jazyk C# značně podobá jazyku Java. C# je stejně jako Java založen na syntaxi jazyka C; nalezneme zde obvyklé třídy, rozhraní a metody, správa paměti je stejně jako v Javě založena na garbage collectoru. Tento fakt vedl k mnohým komentářům o tom, že C# je tupá a nekompatibilní nápodoba Javy, kterou Microsoft vymyslel jenom proto, aby odvrátil zájem vývojářů od Sunu. Při druhém, podrobnějším, pohledu jsou ovšem patrné dosti podstatné rozdíly, které většinou vyznívají ve prospěch C#.

To je způsobeno hlavně tím, že jeho autoři měli jistě výhodu v tom, že za pět let existence Javy mohli nasbírat praktické zkušenosti s programovacím jazykem takového typu.

6.3.3 Objektová orientovanost

C# je silně objektově orientovaný jazyk, který podobně jako Java nahrazuje některé neobjektové prvky jazyků objekty. Namísto globálních proměnných, konstant a funkcí se používají statické prvky tříd, je plně podporováno zapouzdření, polymorfismus a dědění. Všechny metody jsou standardně nevirtuální, což znemožňuje nechtěné předefinování metody, prostě protože jste zapomněli, že už jste ji použili dříve. Lokální metody a operátory mohou být v C# přetěžovány. Ukazatele na funkce jsou nahrazeny delegáty - objektovým ekvivalentem.

6.3.4 Jednoduchost a modernost

Například v C++ je někdy nepříjemné rozlišovat mezi zápisem pro ukazatel (->), člen třídy (::) a kdy použít pouhou tečku. Přestože překladač vždy ví, co na daném místě mělo být správně, tak programátor často musí hledat v kódu, který zápis je správný právě pro určité místo. V C# jsou všechny výše popsané zápisy nahrazeny tečkou. Když pracujete s členem třídy, celou třídou jmenným prostorem nebo ukazatelem není nutné přemýšlet nad tím, co napíšete.

Dále z jazyka částečně mizí ukazatele. Například parametry předávané odkazem se již nepředávají jako ukazatele. Ukazatele je však za určitých předpokladů možné používat i nadále.

Další zjednodušení jsou např. v případě rozhodovacího bloku switch, či při použití integeru pro práci s logikou.

C# sjednocuje systém typů v jazyce tím, že se s každým typem zachází jako s objektem. Objekty jsou kombinovány v jmenné prostory (namespace), které umožňují ke všemu přistupovat přímo z kódu. Ve výsledku to vede k zjednodušení práce.

V COM+ všechny třídy existují v jednom, hierarchicky setříděném jmenném prostoru. V C# pak při použití příkazu using není nutné specifikovat celou cestu, když používáte určitou metodu třídy patřící do daného jmenného prostoru. Například jmenný prostor System obsahuje několik tříd, jako třeba třídu Console, která má metodu WriteLine, která samozřejmě vypíše řádek textu na textovou konzoly. Takže nemusíme psát: System.Console.WriteLine() ale stačí pouze Console.WriteLine().

I již zmiňovaný Garbage collection je jedním z příkladů moderních prvků. Cokoliv, co na sebe ztratí odkaz je uvolněno z paměti. I obsluha chyb je o proti C/C++ pojata v moderním duchu přes prvky try ... catch a try ... finally.

Pravda je, že i toho lze v C++ docílit přes použití maker, ale zde je to již začleněno do jazyka.

Podpora typů v jazyce C# je více zaměřena na data pocházející z reality, objevují se typy decimal a string. Navíc je umožněno vytváření vlastních typů, které se chovají stejně efektivně, jako ty implicitní.

Významnou vlastností C# je tzv. sjednocení typů. Libovolný typ lze, bez ohledu na to, zda je hodnotový, nebo odkazový, převést na třídu i object a zpět. Zpětný převod se pochopitelně provede pouze v případě, že proměnná typu object obsahuje hodnotu (nebo odkaz v případě, že byl přiřazen referenční typ) požadovaného typu (jinak dojde k výjimce).

Převodu hodnotového typu na objekt sémanticky odpovídá vytvoření nové instance třídy, která obsahuje jediný atribut požadovaného typu a přiřazení hodnoty do tohoto atributu. Schopnost pracovat s hodnotovými typy jako s objekty tak vyplňuje mezeru, která existuje ve většině objektově orientovaných jazyků, jako je Java nebo Objective C. Například kontejnerové typy (pole, zřetězené seznamy apod.) stačí implementovat pro instance třídy object, a přitom je lze bez dalšího úsilí používat pro skladování základních typů nebo záznamů.

K moderním prvkům patří i podpora odlaďovací fáze programu. V C++ použití instrukcí #ifdefs vede nakonec k vytvoření kódu, který obsahuje volání mnoha funkcí, které nedělají vůbec nic. C# nabízí pro tyto účely klíčové slovo conditional, které lehce vytváří funkce pouze pro debugging programu.

6.3.5 Typová bezpečnost

Její přítomnost v programovacím jazyce je předpokladem opravdové robustnosti výsledného kódu.

Inicializace proměnných

Všechny dynamicky alokované objekty a pole jsou po vytvoření snulovány. To se sice netýká lokálních proměnných, ale kompilátor vypisuje varování, pokud takovou proměnnou použijete před její inicializací.

Kontrola přetečení

Výsledky všech aritmetických operací v jazyce C# jsou hlídány proti přetečení rozsahu daného typem proměnné. Stejně tak jsou kontrolovány převody mezi jednotlivými typy. Kontrolu přetečení je však možné explicitně vypnout, pokud je to záměrem programátora.

Práce s poli

Pole jsou v jazyce C# pojaty jako objekty, které obsahují odkazy na své hodnoty. Deklarace polí umožňuje vytvářet všechny myslitelné druhy polí - vícerozměrné i vnořené (pole polí).

Protože jsou pole chápány jako objekty, tak po deklaraci nemají konkrétní rozměr. Ten je jim přiřazen až při vytvoření konkrétní instance. S vnořenými poli je to o trochu méně pohodlné. Pole je možné v deklaraci a vytvoření instance přímo inicializovat.

Protože jsou pole objekty odvozené od třídy System.Array, je na nich možné vyvolávat jejich metody, které je umožní setřídit, přistupovat k jednotlivým prvkům nebo v nich vyhledávat.

6.3.6 Škálovatelnost

Když zdrojový kód v C# naroste je možné ho rozdělit do jednotlivých souborů. Není nutné sledovat kde je která hlavička nebo kam patří která rutina. Soubory se zdrojovým kódem je možné slučovat, rozdělovat a přejmenovávat dle libosti. Kompilátor si stále udržuje přehled, kde co najde. Vyplývá to z toho, že se pracuje s jmennými prostory a ne s jednotlivými soubory.

6.3.7 Flexibilita

Z výše uvedeného vyplývá, že C# má mnoho vlastností, které dělají kód bezpečnější tím, že odstraňují některé části nebezpečné volnosti v C++. C# ale pro určité aplikace umožňuje tyto omezení odstranit vytvořením nezabezpečených tříd, kde je možné používat ukazatele, struktury a statická pole. Zde se nebude používat typová bezpečnost, ale budou odděleny od bezpečného kódu a nebude garantována funkce garbage collection.

6.4 Microsoft Access

Již letitý (poprvé uveden v roce 1992) „člen rodiny“ Microsoft Office asi není třeba moc představovat. V porovnání s „profesionálnějšími“ konkurenty typu Microsoft SQL, MySQL, DB/2, Oracle,… jde o jednodušší, univerzální, intuitivní (mohou ji využít ti nejzkušenější uživatelé databází, ale i uživatelé pracující s databázemi poprvé) a relativně levnou aplikaci, která nabízí účinné nástroje pro správu i analýzu dat, ale, díky své menší robustnosti, flexibilitě a rychlosti, určenou spíše pro méně rozsáhlé projekty.

Nejnovější verze – Microsoft Access 2002 z kancelářského sady Microsoft Office XP, poskytuje nástroje pro vytváření řešení, která díky integraci a využití standardů sítě Internet, jako jsou například formáty XML, XSL a dynamické webové stránky, umožňují lepší sdílení a prezentaci dat v sítích intranet a Internet. Obsahuje nové funkce, jako například kontingenční tabulku a kontingenční graf, které výrazně zdokonalují možnosti analýzy dat.

6.5 ADO .NET

6.5.1 Úvod k ADO.NET

ADO.NET (ActiveX Data Objects) je základní technologií pro přístup k datům a manipulaci s nimi v prostředí Microsoft .NET. ADO.NET zajišťuje konzistentní přístup k datovým zdrojům jako je Microsoft SQL Server

nebo data databází OLE DB a XML. Uživatelské aplikace sdílející data mohou použít ADO.NET pro spojení k těmto datovým zdrojům, získávání údajů, zobrazování a manipulaci s nimi.

Do ADO.NET patří .NET poskytovatelé dat (providers) zajišťující spojení na databáze, provedení příkazů a získávání výsledků. Výsledky jsou zpracovány buď přímo nebo jsou umístěny do objektu DataSet, ve kterém může uživatel kombinovat různé zdroje nebo přesouvat data mezi vrstvami. Objekt DataSet lze také použít nezávisle na .NET poskytovali dat a spravovat lokální data aplikace nebo data XML.

6.5.2 Proč ADO.NET?

Způsob přípravy nových aplikací se mění, nové aplikace stále více směřují do prostředí Webu. Rostoucí počet aplikací používá XML pro kódování dat posílaných po síti. Web aplikace využívají HTTP jako nástroj pro komunikaci mezi vrstvami a proto musí ošetřovat správu stavu mezi jednotlivými požadavky. Nový model se výrazně liší od trvale propojeného programovacího stylu, který charakterizoval éru klient/serveru. Základem bylo trvale otevřené spojení k datovému zdroji během celé práce aplikace a nebylo proto nutno řešit otázky spojené se správou stavu.

ADO.NET je navrženo taky, aby vyhovělo potřebám nového programového modelu: odpojené datové architektuře, úzkému propojení s XML, společné datové reprezentaci s možností kombinace datových zdrojů a optimalizací služeb pro interakci s databázemi, všechno jako součást .NET Framework.

6.5.3 Hlavní výhody

Využití stávající znalosti ADO

Návrh ADO.NET reaguje na mnoho požadavků současného ADO programovacího modelu. Programovací model ADO.NET je podobný ADO a proto současní programátoři ADO nemusí při studiu nové technologie začínat s výukou od samého začátku. ADO.NET je vnitřní součástí .NET Framework.

ADO.NET existuje současně s ADO. ADO je i nadále v .NET k dispozici prostřednictvím služeb .NET COM.

Podpora pro vícevrstvý programovací model

ADO.NET zajišťuje prvotřídní podporu pro odpojené, vícevrstvé programové prostředí, ve kterém budou navrhovány budoucí aplikace. Koncept práce s odpojenou sadou záznamů se stal základem nového programového modelu. Řešením ADO.NET pro vícevrstvé programování je použití objektu DataSet.

Podpora XML

XML a datový přístup spolu velmi úzce souvisí—XML je všechno o kódování dat a pro přístup k datům se stále více uplatňuje XML. .NET Framework Web standardy pouze nepodporuje - ony tvoří základní součásti jeho modelu. Podpora XML je zakotvena v ADO.NET na základní úrovni. XML class framework v .NET a ADO.NET jsou součástí stejné architektury — jsou integrovány do mnoha různých úrovní. Již není nutné volit mezi sadou služeb pro datový přístup a jejich XML součástmi; schopnost snadného přechodu mezi nimi navzájem je zařazena do návrhu obou.

6.5.4 Architektura ADO.NET

V centru softwarového řešení, které používá ADO.NET je objekt Dataset. Dataset je kopie databázových dat v paměti. Obsahuje libovolný počet tabulek, z nichž každá odpovídá tabulce nebo pohledu v databázi. Dataset vytváří odpojený pohled na data databáze. Znamená to, že existuje v paměti bez aktivního spojení k databázi, která obsahuje odpovídající tabulky a pohledy. Odpojená architektura zajišťuje při čtení nebo psaní z/do databáze větší škálovatelnost při pouhém použití zdrojů databázového serveru.

Data jsou předávána z databáze objektům střední vrstvy a dolů na rozhraní uživatele. Pro shromáždění datových změn používá ADO.NET přenosový formát XML. Znamená to, že pro přenos dat mezi vrstvami řešení ADO.NET

transformuje v paměti data (dataset) do XML a potom je jako XML předává jiné komponentě.

6.5.5 Poskytovatelé dat pro .NET

Poskytovatele dat .NET (data provider) použijete, když chcete specifikovat spojení k databázi, provést příkazy a získat výsledky. Výsledky můžete zpracovat přímo nebo prostřednictvím ADO.NET DataSet. Data provider .NET je navržen tak, aby minimalizoval náročnost přenosu dát mezi datovým zdrojem a kódem a zvyšoval výkonnost a neztrácel na funkčnosti.

.NET Framework nabízí SQL Server .NET Data Provider (pro Microsoft SQL Server 7.0 a pozdější verze) a OLE DB .NET Data Provider.

6.5.6 SQL Server .NET Data Provider

SQL Server .NET Data Provider používá vlastní protokol pro komunikaci s SQL Server. SQL Server .NET Data Provider je jednoduchý a zajišťuje přístup k datům SQL Serveru bez doplňování OLE DB nebo vrstvy Open Database Connectivity (ODBC).

SQL Server .NET Data Provider lze použit pouze pro spojení s Microsoft SQL Server 7.0 nebo pozdější. Třídy SQL Server .NET Data Provideru se nacházejí v namespace System.Data.SqlClient . Pro starší verze Microsoft SQL Serveru je nutné použit OLE DB .NET Data Provider s SQL Server OLE DB Provider (SQLOLEDB).

Při použití SQL Server .NET Data Provider, je nutné zahrnout do aplikace namespace System.Data.SqlClient .

6.5.7 OLE DB .NET Data Provider

OLE DB .NET Data Provider používá pro přístup k datům nativní OLE DB prostřednictvím COM. Podporuje jak manuální, tak i automatické

transakce. Automatické transakce OLE DB .NET Data Provider automaticky organizuje do transakcí a získává detailní transakce z Windows 2000 Component Services.

Při použití OLE DB .NET Data Provider, je nutné zahrnout do aplikace namespace System.Data.OleDb .

6.5.8 Výběr poskytovatele dat pro .NET

Výběr poskytovatele dat pro aplikaci ovlivní výkon, možnosti a integritu výsledné aplikace. Proto je nutné zvolit poskytovatele přesně podle návrhu a datového zdroje.

SQL Server .NET Data Provider

Doporučuje se:

- pro střední vrstvu aplikací používající Microsoft SQL Server 7.0 nebo pozdější verze.
- pro jednovrstvé aplikace používající Microsoft Data Engine (MSDE) nebo Microsoft SQL Server 7.0 nebo pozdější verze.
- přednost před použitím OLE DB Provider pro SQL Server (SQLOLEDB) s OLE DB .NET Data Provider.

Pro Microsoft SQL Server 6.5 a starší verze je nutné použít OLE DB Provider pro SQL Server s OLE DB .NET Data Provider.

OLE DB .NET Data Provider

Doporučuje se:

- pro střední vrstvu aplikací používající Microsoft SQL Server 6.5 nebo starší verze nebo Oracle.
- pro jednovrstvé aplikace používající databáze Microsoft Access.

Použití OLE DB .NET Data Provider s Microsoft Access databází pro střední vrstvu není doporučeno. Podpora pro OLE DB Provider pro ODBC se neposkytuje.

7 Závěr

Výsledná aplikace se značně liší od prvních návrhů, a to jak ve struktuře databáze, tak i uživatelském rozhraní. U databázové struktury to bylo z důvodu postupu od jednoduššího modelu ke složitějšímu, zatímco u uživatelského rozhraní to bylo hlavně z důvodu postupného seznamování se s jednotlivými ovládacími prvky a jejich vlastnostmi. Konečná verze je pak jakýmsi kompromisem mezi jednoduchostí a snadným ovládáním aplikace, vlastnostmi a možnostmi ovládacích prvků a strukturou databáze.

Aplikace je založená na konkrétních požadavcích, určená pro konkrétní firmu a plně nahrazuje stávající informační systém pro správu skladového hospodářství. Pro obecné využití by bylo nutné některých úprav, které by postihly obecnější požadavky na informační systémy tohoto druhu.

Výhodou i nevýhodou aplikace je to, že je vytvořena pro platformu Microsoft .NET. Mezi výhody lze počítat podstatně menší konečnou velikost souboru, či snadná instalace a odinstalace programu, tedy pouhé zkopírování do cílového adresáře resp. pouhé smazání souborů aplikace. Na druhé straně jsou tu nevýhody v podobě náročnějších požadavků na systémové prostředky, či menší rychlost programu.

8 Použitá literatura

[1] Chand, M. (2002): A Programmer‘s Guide to ADO.NET in C#, Apress, Berkley

[2] Troelsen, A. (2001): C# and the .NET Platform, Apress, Berkley

[3] Kačmář, D. (2001): Programujeme .NET aplikace ve Visual Studiu .NET, Computer Press, Praha

[4] Fox, D. (2002): Naučte se ADO.NET za 21 dní, Computer Press, Praha

PŘÍLOHY

Seznam příloh

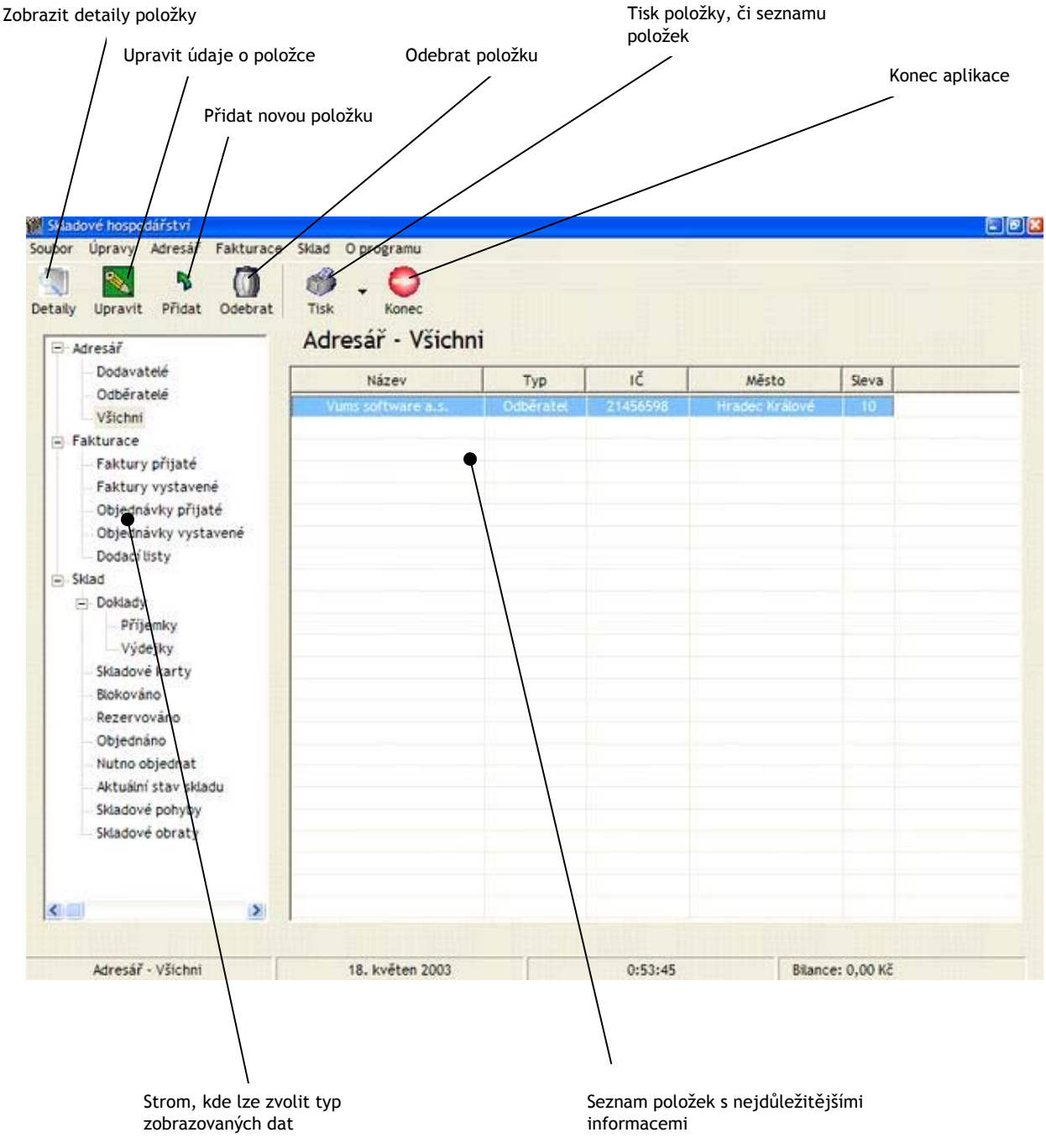
1. Jednoduchý návod pro obsluhu
2. Zdrojové texty aplikace – na přiloženém CD-ROMu

Příloha číslo 1: Jednoduchý návod pro obsluhu

Instalace a spuštění programu

Celá instalace se skládá pouze ze zkopírování souboru Sklad.exe společně s databázovým souborem (databaze.mdb) a se soubory dodacilist.frm, faktura.frm a objednavka.frm do libovolného adresáře. Start aplikace se pak provede spuštěním souboru Sklad.exe. Pro spuštění aplikace je potřeba mít nainstalovaný Microsoft .NET Framework.

Obsluha programu



Zobrazovaná data lze volit jednak pomocí stromu a jednak pomocí hlavního menu. Operace s položkami lze volit přes tlačítka ovládacího panelu nebo přes hlavní menu.

Zobrazovaná data jsou rozdělena do tří tématických skupin:

Adresář

Tato sekce obsahuje seznam dodavatelů a odběratelů a odběratelů a dodavatelů dohromady.

Položky lze snadno přidávat i odebírat, upravovat a vytisknout seznam odběratelů, dodavatelů, či společný seznam odběratelů a dodavatelů.

Fakturace

Tato část obsahuje seznamy dokumentů jako jsou přijaté a vystavené objednávky, přijaté a vystavené faktury a dodacích listů.

Jednotlivé dokumenty lze samozřejmě opět přidávat, odebírat, upravovat i tisknout a to, jak seznamy dokumentů, tak jednotlivé dokumenty. Navíc se vybavením objednávky, tedy změnou stavu z nevybavené na vybavenou odpovídajícím tlačítkem, automaticky vygeneruje faktura a dodací list. Jednotlivé dokumenty lze skrýt tak, aby se nezobrazovaly v seznamu dokumentů a ani netiskly.

Sklad

Tato část obsahuje jednak seznam skladových dokladů, tedy příjemek a výdejek, skladových karet a pak seznamy zboží roztříděné podle různých filtrů tedy zboží, které je blokováno, rezervováno, objednáno a nutno objednat, aktuální stav skladu, skladové obraty za určité období a pohyby na skladě.

Skladové doklady a skladové karty lze opět přidávat, odebírat, upravovat a tisknout seznamy, stejně jako skrývat.

Hlavní menu

Hlavní menu je rozděleno do šesti částí. Menu „Adresář“, „Fakturace“ a „Sklad“ odpovídají stejně nazvaným sekcím ve stromu, tedy lze jimi volit typ zobrazovaných dat.

Menu „Soubor“ umožňuje export adresáře a faktur, náhled tisku, tisk a ukončení aplikace.

Menu „Úpravy“ odpovídá tlačítkům ovládacího panelu (detaily, přidat, upravit a odebrat), tedy lze volit operace s položkami, jako je přidání nové položky, detaily, upravení, či odebrání zvolené položky.

Menu „O programu“ obsahuje krátkou informaci o programu a o autorovi.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č.121/2000 o právu autorském, zejména § 60 (školní dílo) a § 35 (o nevýdělečném užití díla k vnitřní potřebě školy).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

V Liberci 23. 5. 2003

…….

Martin Hudík